CLAIMS

1. A computing environment wherein a process comprising a set of data and/or program modules and execution state is treated as an entity that can be transferred between components of said environment, and wherein a said process can be subject to an evolutionary operation.

2. A computing environment as claimed in claim 1 wherein a construct is formed comprising data and/or program modules and execution state of a first process, and wherein said evolutionary operations are performed by functions operating on a said construct.

3. A computing environment as claimed in claim 2 wherein said construct is formed by a construct operation that suspends all active threads of said first process and creates a new process comprising at least some of the data and/or program modules and execution state of said first process, and stores said new process in a data area of said first process.

4. A computing environment as claimed in claim 3 wherein said construct comprises only data, program modules and execution state falling within lists that are passed to said construct operation.

5. A computing environment as claimed in any of claims 2 to 4 wherein said construct is provided with an authorising signature.

6. A computing environment as claimed in any preceding claim wherein said evolutionary operations include the selective deletion of objects from within said process.

7. A computing environment as claimed in any preceding claim wherein said evolutionary operations include the selective loading or reloading of objects into said process.

8. A computing environment as claimed in claim 1 wherein a said evolutionary operation includes the incorporation into a first process of new objects from a second process.

9. A computing environment as claimed in claim 8 wherein a construct is formed comprising at least some of the data and/or program modules and execution state from said second process, and said construct is transferred to said first process.

10. A computing environment as claimed in claim 9 wherein said construct is formed by a construct operation that suspends all active threads of said second process and creates a new process comprising a subset of the data, program modules and execution state of said second process, and stores said new process in a data area of said second process.

11. A computing environment as claimed in claim 10 wherein said construct comprises only data, program modules and execution state falling within lists that are passed to said construct operation.

12. A computing environment as claimed in any of claims 9 to 11 wherein said construct is provided with an authorising signature.

13. A computing environment as claimed in any of claims 9 to 12 wherein after said construct is transferred the second process stored within said construct is caused to be activated within said first process.

14. A computing environment as claimed in any of claims 9 to 12 wherein after said construct is transferred the first process is suspended and the second process stored within said construct is activated, and when the second process is concluded the data and program modules of the second process are added to the first process and the first process is re-activated.

15. A computing environment as claimed in any of claims 9 to 12 wherein after said first process terminates at least some of the data and/or program modules from said first process are added to the second process stored in said construct and said second process is then activated.

16. A computing environment as claimed in claim 8 wherein a construct is formed comprising at least some of the data and/or program modules from said second process, and said construct is transferred to said first process.

17. A computing environment as claimed in claim 16 wherein said construct is formed by a construct operation that suspends all active threads of said second process and creates a new process comprising at least some of the data and/or program modules of said second process, and stores said new process in a data area of said second process.

18. A computing environment as claimed in claim 17 wherein said construct comprises only a subset of data and program modules falling within lists that are passed to said construct operation.

19. A computing environment as claimed in any of claims 16 to 18 wherein said data and said program modules from said second process are copied into said first process.

20. A computing environment as claimed in any of claims 8 to 19 wherein in the event of a conflict between data and/or program modules of said first process and data and/or program modules of said second process, the data and/or program modules of said first process will override the data and/or program modules of said second process.

21. A computing environment as claimed in any of claims 8 to 19 wherein in the event of a conflict between data and/or program modules of said first process and data and/or program modules of said second process, the data and/or program modules of said second process will override the data and/or program modules of said first process.

22. A computing environment as claimed in claim 1 wherein a said process may be transferred between different first and second hardware components of said computing environment.

23. A computing environment as claimed in claim 22 wherein a construct is formed comprising at least some of the data and/or program modules and execution state of said process, and said construct is transferred.

24. A computing environment as claimed in claim 22 wherein said construct comprises a subset of the data, program modules and execution state of said process.

25. A computing environment as claimed in any of claims 22 to 24 wherein a said process is subject to an evolutionary operation that allows the process to run in the second hardware component.

26. A computing environment as claimed in any of claims 22 to 24 wherein said second hardware component is a memory storage device.

27. A method for controlling the evolution and migration of a process comprising a set of data and program modules and execution state within a computing environment, wherein a construct is formed comprising data and/or program modules and execution state of a first process.

28. A method as claimed in claim 27 wherein said construct is formed by a construct operation that suspends all active threads of said first process and creates a new process comprising at least some of the data and/or program modules and execution state of said first process, and stores said new process in a data area of said first process.

29. A method as claimed in claim 28 wherein the construct comprises all the data, program modules and execution states of said first process.

30. A method as claimed in claim 28 wherein said construct comprises only data, program modules and execution state falling within lists that are passed to said construct operation.

31. A method as claimed in any of claims 28 to 30 wherein said construct is provided with an authorising signature.

32. A method as claimed in any of claims 27 to 31 wherein a process is modified by the selective deletion of objects from within the process.

33. A method as claimed in any of claims 27 to 31 wherein a process is modified by the selective loading or reloading of objects into a said process.

34. A method as claimed in claim 27 comprising incorporating into a first process objects from a second process.

35. A method as claimed in claim 34 wherein a construct is formed comprising at least some of the data and/or program modules and execution state from said second process, and said construct is transferred to said first process.

36. A method as claimed in claim 35 wherein said construct is formed by a construct operation that suspends all active threads of said second process and creates a new process comprising some of the data and/or program modules and execution state of said second process, and stores said new process in a data area of said second process.

37. A method as claimed in claim 36 wherein said construct comprises only data, program modules and execution states falling within lists that are passed to said construct operation.

38. A method as claimed in any of claims 35 to 37 wherein said construct is formed with an authorising signature.

39. A method as claimed in any of claims 35 to 38 wherein after said construct is transferred the second process stored within said construct is caused to be activated within said first process.

40. A method as claimed in any of claims 35 to 38 wherein after said construct is transferred the first process is suspended and the second process stored within said construct is activated, and when the second process is concluded the data and program modules of the second process are added to the first process and the first process is re-activated.

41. A method as claimed in any of claims 35 to 38 wherein after said first process terminates at least some of the data and/or program modules from said first process are added to the second process stored in said construct and said second process is then activated.

42. A method as claimed in claim 34 wherein a construct is formed comprising at least some of the ~~data and/or~~ program modules from said second process, and said construct is transferred to said first process.

43. A method as claimed in claim 42 wherein said construct is formed by a construct operation that suspends all active threads of said second process and creates a new process comprising some of the data and program modules of said second process, and stores said new process in a data area of said second process.

44. A method as claimed in claim 43 wherein said construct comprises only data and program modules falling within lists that are passed to said construct operation.

45. A method as claimed in any of claims 42 to 44 wherein said at least some data and/or program modules from said second process are copied into said first process.

46. A method as claimed in any of claims 34 to 45 wherein in the event of a conflict between data and/or program modules of said first process and data and/or program modules of said second process, the data and/or program modules of said first process will override the data and/or program modules of said second process.

47. A method as claimed in any of claims 34 to 45 wherein in the event of a conflict between data and/or program modules of said first process and data and/or program modules of said second process, the data and/or program modules of said second process will override the data and/or program modules of said first process.

48. A method as claimed in claim 27 wherein a said process is transferred between different first and second hardware components of said computing environment.

49. A method as claimed in claim 48 wherein a construct is formed comprising at least some of the data and/or program modules and execution state of said process, and said construct is transferred.

50. A method as claimed in claim 49 wherein said construct comprises a subset of the data, program modules and execution state of said process.

51. A method as claimed in any of claims 48 to 50 wherein a said process is subject to an evolutionary operation that allows the process to run in the second hardware component.

52. A method as claimed in any of claims 48 to 50 wherein said second hardware component is a memory storage device.

Add
Cl